

The Bioinformatics Lab: Web Server

Anita-Juliane Winkler

June 7, 2010

Chapter 1

Introduction

The programming challenge for May 31st, 2010 was to install and configure our own apache webserver (Apache2) and to enable SSL-encrypted communication (HTTPS). Furthermore Apache was extended by debian modules enabling PHP. Examining different ways of access control, we tested our application by implementing a simple private homepage with PHP giving only access to people in our own network.

1.1 Web Server

A web server is a computer program that delivers content over a browser (World Wide Web) using the Hypertext Transfer Protocol (HTTP). The primary function of a web server is to deliver web pages to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and JavaScripts. A client, commonly a web browser, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource, which is typically a real file on the server's secondary memory.

Many generic web servers also support server-side scripting, such as Apache HTTP Server. The server-side scripts, written in e.g. PHP, are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. Usually, the user cannot see the script's source code and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts, which run by the viewing web browser, usually written in JavaScript.

Web servers are able to map the path component of a Uniform Resource Locator (URL) into either a local file system resource (for static requests) or an internal or external program name (for dynamic requests). For a static request the URL path specified by the client is relative to the Web server's root directory. The Web server on the host will append the given path to the path of its root directory (`/var/www`), which will result in a local file system resource. The file will be read and a response (file content) will be send to the client's Web browser.

1.2 Apache

Since 1996 the Apache HTTP Server, commonly referred to as Apache, is the most popular web server software in use, which evolved to rival other Unix-based web servers in terms of functionality and performance. With the use of over 60% of all web servers, Apache serves as the most popular software. Released under the Apache License, Apache is characterized as open source software, which is available on <http://httpd.apache.org/>.

This versatile and high-performance HTTP server is designed to reduce latency and increase throughput, relative to simply handling more requests. This ensures consistent and reliable processing of requests within reasonable time-frames.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support, such as `mod_php`, which embeds a PHP interpreter into Apache, to authentication schemes, such as `mod_access`. Another optional module includes a Transport Layer Security (TLS) and Secure Socket Layer (SSL) support (`mod_ssl`). TLS and its predecessor SSL are cryptographic protocols that provide security for communications over networks such as the Internet. Virtual hosting allows one Apache installation to serve many different actual websites.

Apache is primarily used to serve both static content and dynamic web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides. Apache is used for many other tasks where content needs to be made available in a secure and reliable way. One example is sharing files from a personal computer over the Internet. A user who has Apache installed on their desktop can put arbitrary files in Apache's document root which can then be shared.

The latest release of the Apache HTTP Server ("httpd") is version 2.2.15, which is principally a security and bugfix release. New features allow this next generation web server to be scalable and extendable making it to the most stable version of Apache HTTP Server.

1.3 HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is a combination of the HTTP with the SSL/TLS protocol to provide encryption and secure identification of the server by website security testing. For HTTP communication the browser creates a SSL connection to the Transmission Control Protocol (TCP)-Port of the web server. HTTPS use port 443 by default and URLs begin with "`https://`", opposed to HTTP URLs, which start with "`http://`" and use port 80 by default. While HTTP is insecure and as a consequence lets attackers gain access to website accounts and sensitive information, HTTPS is designed to withstand such eavesdropping attacks and is considered secure. HTTPS is not a separate protocol, but refers to the use of ordinary HTTP over an encrypted SSL or TLS connection, which is good for privacy, integrity and authenticity.

The system can also be used for client authentication in order to limit access to a web server to authorized users. To do this, the site administrator typically creates a certificate for each user, which is loaded into a browser. Normally,

that contains the name and e-mail address of the authorized user and is automatically checked by the server on each reconnect to verify the user's identity, potentially without even entering a password.

Chapter 2

Programming Challenge

Within this chapter the single steps for setting up an own web server will be represented. Before performing the required steps, it is recommended to work within the graphical user interface by starting up the X server (`startx`) as well as to update your package list (`apt-get update`) and upgrade your packages (`apt-get upgrade`).

2.1 Installation and Configuration of Apache

In the first step install the Apache2 package using following command:

```
sudo apt-get install apache2.
```

A successful installation can be verified by testing Apache at the localhost. For this step use any available browser, e.g. Iceweasel: `http://localhost` should work and now you have the possibility to change the default content of the `index.html` file, which can be found in the `/var/www` directory.

Getting used to the Apache process manager `/etc/init.d/apache2`, you should always reload new configurations using the `reload` option. In order to restart and terminate background processes (daemon/demon), signals should be send to the parent processor. The option `status` tells you at which process ID the Apache process is running.

2.1.1 Configuration File

The default configurations can be found in `/etc/apache2`. Within this direcorey you can find the configuration file `apache2.conf`, which gives you useful information about u.a. the error log, which goes to `/var/log/apache2/error.log`. As another example serve the `/etc/apache2/mods-enabled/*.conf` files that include module's configurations.

Modules

Information on available module packages can be achieved by following commands:

- `a2enmod` script that enables the specified module within the Apache2 configuration by creating symbolic links within `/etc/apache2/mods-enabled`-this will cause them to be actually loaded/read

- `a2dismod` script that disables a module by removing those symbolic links

Virtual Host

In order to access our own website with our hostname a virtual host is created, which hosts multiple domain names using a single IP address. For this step the `a2ensite` script is necessary which enables the specific site within the Apache2 configuration by creating symbolic links within `/etc/apache2/sites-available`. The default site is handled specially: The resulting symbolic link will be called `000-default` in order to be loaded first. Delete this enabled default file to avoid a failed request.

Create a file (`vim /etc/apache2/sites-available/xxx`) containing information on our available site by adding following lines:

```
NameVirtualHost *
<VirtualHost *>
ServerName name
ServerAlias hostname
DocumentRoot /var/www
ServerAdmin webmaster@hostname
# Logfiles:
CustomLog /var/log/apache2/access2.log combined
ErrorLog /var/log/apache2/error2.log
LogLevel warn
<Location />
Options Indexes FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
allow from all
</Location>
</VirtualHost>
```

The description of each directive are listed as follows:

- `NameVirtualHost`: designates an IP address for name-virtual hosting
- `<VirtualHost>`: contains directives that apply only to a specific hostname or IP address
- `ServerName`: hostname and port that the server uses to identify itself
- `ServerAlias`: alternate names for a host used when matching requests to name-virtual hosts
- `DocumentRoot`: forms the main document tree visible from the web
- `ServerAdmin`: email address that the server includes in error messages sent to the client
- `CustomLog`: sets filename and format of log file
- `ErrorLog`: location where the server will log errors
- `LogLevel`: controls the verbosity of the `ErrorLog`

- **<Location>**: applies the enclosed directives only to matching URLs

Using the **NameVirtualHost** directive you can configure the designation of the IP address (and possibly port) on the server that will be accepting requests for the hosts. Using ***** as the argument any and all IP addresses are used on the server. If you want to use multiple ports, add it to the argument, such as ***:80**. A **<VirtualHost>** block is created for each different host to be served. Note that the argument to the **<VirtualHost>** directive must match a defined **NameVirtualHost** directive.

In much the same way that modules are enabled and disabled, sites (virtual hosts) are kept in **/etc/apache2/sites-available** and symlinked to **/etc/apache2/sites-enabled** with the **a2ensite** and **a2dissite** commands. After enabling the virtual host by **a2ensite xxx**, reload the new configuration.

HTTPS

First of all we need a server certificate. For this step enable the SSL module by **a2enmod ssl** and perform following command in the previously created **/etc/apache2/ssl** directory:

```
openssl req -new -x509 -nodes -out hostname.crt -keyout hostname.key.
```

The certificate is written in the **.crt** file and the **.key** file contains the newly created private key.

Instead of paying any costs for the authentication of our certificate from an official organization, we will accept any occurring warnings and import the certificate at the browser start. CAcert.org is one certification unit, that issues certificates to the public for free. To start the SSL engine, we have to adjust our virtual host by adding following lines:

```
# SSL
SSLEngine On
SSLCipherSuite HIGH:MEDIUM
SSLCertificateFile /etc/apache2/ssl/hostname.crt
SSLCertificateKeyFile /etc/apache2/ssl/hostname.key
```

The previously generated files are Privacy Enhanced Mail (PEM)-encoded and used by following directives:

- **SSLCertificateFile**: server certificate
- **SSLCertificateKeyFile**: private key of the certificate

The PEM file type is primarily associated with 'Privacy Enhanced Mail Security Certificate'. In cryptography, a public key certificate (or identity certificate) is a certificate, that uses a digital signature to bind together a public key with an identity. The certificate can be used to verify that a public key belongs to an individual. PEM, is an early proposal for securing email using public key cryptography, which was never widely deployed or used.

In the end, Apache is restarted to save the add-ons.

2.2 Enabling PHP

Hypertext Preprocessor (PHP) is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. As a general-purpose programming language, PHP code is processed by an interpreter application in command-line mode performing desired operating system operations and producing program output on its standard output channel. It may also function as a graphical application. PHP is available as a processor for most modern web servers and as standalone interpreter on most operating systems and computing platforms.

When a user navigates in her/his browser to a page that ends with a `.php` extension, the request is sent to a web server, which directs the request to the PHP interpreter. The PHP interpreter processes the page, communicating with file systems, databases, and email servers as necessary, and then delivers a web page to the web server to return to the browser.

Installation of PHP

There exist many modules to extend the Apache web server. `aptitude search apache2-mod-` lists you all available Debian Apache modules. Among them you can find the PHP5 module, which is installed by the `apt-get install libapache2-mod-php5` command. Note that the PHP5 module is not compatible with the multithreaded version of Apache (worker-MPM), where one process can handle many requests. It is only working with the slower standard prefork Multi-Processing-Modul (MPM), which implements a non-threaded, pre-forking web server. Since we are using Debian packages, we only have to perform following command, using the `phpinfo` function for testing the installation:

```
echo '<?php phpinfo() ?>' » /var/www/test.php.
```

The code is stored in the `test.php` file and can be opened in our server. `http://localhost/test.php` shows a table with all known details about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment, OS version information, paths, HTTP headers, and many more.

2.3 Enabling Access Control

There exist various ways to control the access listed in the following:

IP-Address

The `mod_access` module provides access control based on client hostname, IP address, or other characteristics of the client request. Its `Allow` directive affects which hosts can access an area of the server. The first argument to this directive is always `from` followed by subsequent arguments. If `Allow from all` is specified, then all hosts are allowed access. To allow only particular hosts or groups of hosts to access the server, the host can be specified in any certain format. For this purpose the virtual host file is changed and `Allow from all` will be commented out in the original file, naturally:

```
# Single IP-adress:
Allow from 192.168.16.5 192.168.16.15
# IP-network:
Allow from 192.168.16.
Allow from 10.1.0.0/16
Allow from 10.2.0.0/255.255.0.0
# all computer in a DNS-Domain:
Allow from informatik.tu-muenchen.de.
```

Password

The Apache `mod_auth` module allows the use of HTTP Basic Authentication to restrict access by looking up users in plain text password and group files. The `AuthType` directive selects the type of user authentication for a directory. To work, it must be accompanied by each of the following directives:

- `AuthName` sets the name of the authorization realm for a directory, which is given to the client so that the user knows which username and password to send
- `AuthUserFile` sets the name of a textual file containing the list of users and passwords for user authentication
- `Require` selects which authenticated users can access a resource.

So, we have to change the `from`-statement in your virtual host file to:

```
AuthType Basic
AuthName "internal"
AuthUserFile /etc/apache2/htpasswd
Require valid-user.
```

After creating the `passwd` file by the `htpasswd -c /etc/apache2/htpasswd Username` command you can type in your personal password. To remove user from this file, just perform the command again and it will automatically overwrite the old user or open the file and remove the user manually. Since we want to define valid users, simply redo the command without the `-c` option, so that a new user can be appended in the file.

It is also possible to enable access control with certificates or LDAP.

To see if one of these options of access control was configured successfully, browse the HTTPS URL `https://localhost/test.php`.